(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
15 November 2001 (15.11.2001)

PCT

(10) International Publication Number
WO 01/86628 A2

(51) International Patent Classification⁷: G10H

(21) International Application Number: PCT/GB01/01985

(22) International Filing Date: 4 May 2001 (04.05.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

0010967.8	5 May 2000 (05.05.2000)	GB
0010969.4	5 May 2000 (05.05.2000)	GB
0011178.1	9 May 2000 (09.05.2000)	GB
0022164.8	11 September 2000 (11.09.2000)	GB
0030833.8	18 December 2000 (18.12.2000)	GB

(71) Applicant *(for all designated States except US)*: **SSEYO LIMITED** [GB/GB]; Highview House, Charles Square, Bracknell, Berkshire RG12 1DF (GB).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **COLE, John, Tim** [GB/GB]; Sseyo Limited, Highview House, Charles Square, Bracknell, Berkshire RG12 1DF (GB). **COLE, Murray, Peter** [GB/GB]; Sseyo Limited, Highview House, Charles Square, Bracknell, Berkshire RG12 1DF (GB). **LEACH, Jeremy, Louis** [GB/GB]; Sseyo Limited, Highview House, Charles Square, Bracknell, Berkshire RG12 1DF (GB). **BLAMPIED, Paul, Alexander** [GB/GB]; Sseyo Limited, Highview House, Charles Square, Bracknell, Berkshire RG12 1DF (GB). **BAREFOOT, Nicholas, John** [GB/GB]; Sseyo Limited, Highview House, Charles Square, Bracknell, Berkshire RG12 1DF (GB).

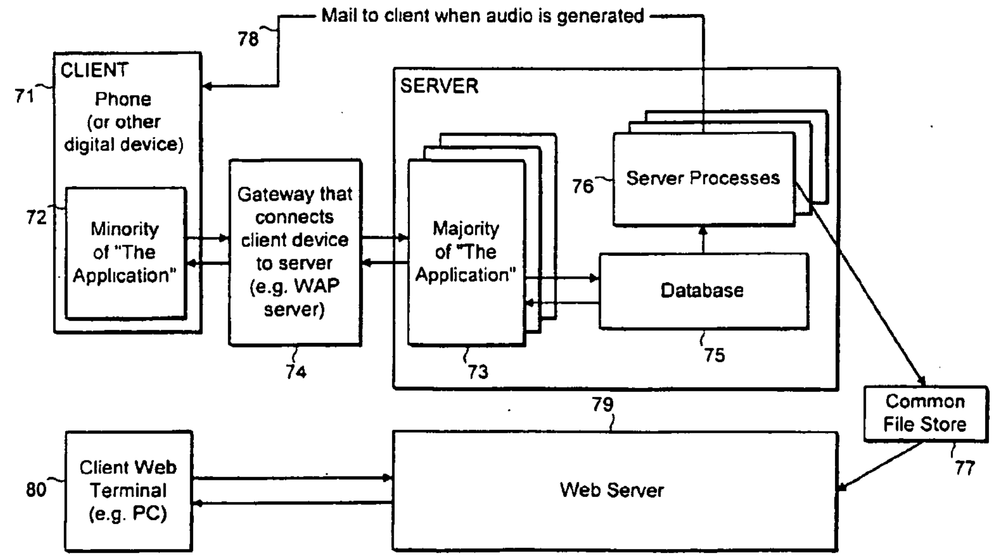
(74) Agents: **MAGGS, Michael, Norman** et al.; Kilburn & Strode, 20 Red Lion Street, London WC1R 4PJ (GB).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE,

[Continued on next page]

(54) Title: AUTOMATED GENERATION OF SOUND SEQUENCES



(57) Abstract: A system for automatically generating sound sequences, for example a generative music system, has a server which communicates with a plurality of remote clients. The generative music system resides entirely on the server, with control coming from the remote clients (71). In some embodiments, the sounds or music generated at the server may depend upon the collaborative or competitive real time actions of the clients. The system has numerous applications, including mobile phones and multi-user computer games.

IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

Automated Generation of Sound Sequences

This invention relates to methods and systems for automated generation of sound sequences, and especially (though not exclusively) of sound sequences in the form of music. It relates in the preferred embodiment to a Generative Music System (GMS).

The automated creation of music has a long history, going back at least as far as Mozart's use of musical dice. One of the first musical works generated by a computer was L. Hiller's Illiac suite. Since that time, of course, the sophistication of computer-generated music or more generally audio sequences has increased substantially.

Systems for creating musical sequences by computer may conveniently be divided up into two areas, which have been called "non-generative" and "generative". Non-generative systems include deterministic systems which will produce the same sequences every time, along with systems that simply replay (perhaps in a random or other order) pre-composed sections of music. The vast majority of current systems which produce musical output make use of this type of approach, for example by selecting and playing a particular predefined sequence of notes at random when the key is pressed or a mouse button clicked. Generative Music Systems, on the other hand, may be considerably more complex. Such systems generate musical content, typically note by note, on the basis of a higher-level of musical knowledge. Such systems either explicitly or implicitly are aware of a variety of musical rules which are used to control or influence the generation of the music. In some systems, the rules may operate purely on the individual notes being generated, without imposing any form of higher order musical structure on the output; in such systems, any musical order that arises will be of an emergent nature. More sophisticated systems may include higher-level rules which can influence the overall musical structure. Generative Music Systems will normally create musical content "on the fly", in other words the musical sequences are built up note by note and phrase by phrase, starting at the beginning and finishing at the end. This means that – in contrast with some of the non-generative systems - the musical content can be generated and played in real time: there is no need for example for the whole of the phrase to be generated before the first few notes of the phrase can be played.

For our present purposes, the essential features of a generative music system are that it generates musical content in a non-deterministic way, based upon a plurality of musical rules (which may either be implicit within the software or which may be explicitly specified by either the program writer or the user of the program). By analogy, a generative sound system produces non-deterministic sound sequences based upon sound-generation rules.

According to a first aspect of the present invention there is provided a system for the composition of audio sequences comprising a server and a generative audio engine on the server controlled by a plurality of remote clients in communication with the server, the server being arranged to transmit a completed composed audio sequence to one or more of the clients. According to a second aspect of the present invention there is provided a method for the composition of audio sequences, comprising composing an audio sequence at a central location, under control of one or more clients at remote locations, and transmitting the completed audio sequence to one or more of the clients.

According to a third aspect of the present invention there is provided a generative audio engine arranged to compose audio sequences under the control of instructions received from remote clients, the engine including means for transmitting a completed audio sequence to one or more of the clients.

The invention further extends to a computer game using a system or method as previously defined.

According to another aspect of the present invention there is provided a method or a system in which composition of a sequence of music or other sounds is effected within, or otherwise in conjunction with, one or more servers.

The method and system of the invention are applicable to the use of generative music (or other sound) systems within a server architecture, and in this regard have many aspects by which such an engine can render audio output. The audio output may be streamed in real-time to networked digital devices, or may be composed on the server as a batch-process for subsequent download to such devices.

The server-based audio generation of the present invention may be arranged to deliver the audio by streaming. The generated audio may in this case be streamed from the generative engine on the server, directly to the client device from an interactive session being conducted by the user. Each stream rendered within the server system might go to a single client, or to a number of clients that all share the same stream. This facilitates server-side creation of generative interactive audio streams, which might be delivered uniquely to one client (for a highly personal audio experience), or shared with multiple clients (for shared audio experiences). Users might be charged for this service.

As an alternative, the download of the server-generated audio work may be deferred. In this case, the generated audio may be saved to audio files (for example, as an MP3 or WAV formatted file) on the server, to be made available to client devices after sufficient server computation time has passed. The download, which may be regarded as providing a recorded generative session, may take place on any client device (for example, a personal computer). More especially, the device receiving the download may not necessarily be the original client device from which the request for audio generation was made. This latter feature is of benefit where there is not sufficient local audio processing power on the original client device (for example, a WAP phone) to accept an audio stream from a server, or perhaps to produce the music in real time. As a general matter, users of the deferred download service, like those of the streaming service, might be charged for using the service.

Networked servers may be used according to invention to generate many different streams and/or files simultaneously. The facility for generation of audio over multiple servers is of advantage in the circumstances in which the amount of audio generation required exceeds the processing power of a single server.

The method and system of the invention may include provision for interaction by clients in creating instructions or rules for generating audio, and in general may be adapted to allow individual clients to manipulate or influence directly the audio that is created on the server. Also, the method and system may adapted to allow group interactive sessions, where many clients share and influence the same composition in either a simultaneous manner, or through deferred processing for later listening by group members.

The general context of the invention is thus that of a generative music or other sound system in which data presented or interpreted in a musical or other sound context is integrated and an output reflecting this integration is generated. The generative system may include multiple interdependent musical or other sound agents that generate a stream of coordinated musical or other sound events which can in turn affect the way in which incoming data is interpreted and integrated, and which can act as an input to processing units within and external to the system. The music or other sound may be delivered either in a real-time generated audio stream, or by being written to a storage medium for deferred delivery. In either case the method and system of the invention may be part of operation and structure of a larger method and system for generating musical works, audio, sounds and sound environments in real-time.

A method and system for automated generation of sound sequences, and applications of such method and system, according to the invention will now be described, by way of example, with reference to the accompanying drawings, in which:

Figure 1 is a schematic representation of an embodiment of the present invention;

Figure 2 is illustrative of objects that are involved in a component of the system of Figure 1;

Figure 3 is illustrative of operation of the method and system of the invention in relation to scale and harmony rules;

Figure 4 illustrates operation of the method and system of the preferred embodiment in relation to the triggering of note sequences and their integration into a musical work as currently being composed and played; and

Figures 5 and 6 illustrate aspects of operation of the method and system of the preferred embodiment in relation to audio generation with rendering at a server.

The method and system to be described are for automated generation of sound sequences and to integrate data presented or interpreted in a musical context for generating an output reflecting this integration. Operation is within the context of generation of musical works, audio, sounds and sound environments in real-time. More especially, the method and system function in the manner of a `generative music system' operating in real-time to enable user-interaction to be incorporated into the composition on-the-fly. The overall construction of the system is shown in Figure 1 and will now be described.

Referring to Figure 1, the system involves four high-level layers, namely, an applications layer I comprising software components 1 to 5, a layer II formed by an application programmer's interface (API) 6 for interfacing with a music (or, more generally, an audio or compositional) engine SKME (SSEYO Koan Music Engine) that is manifest in objects or components 7 to 14 of a layer III, and a hardware device layer IV comprising hardware components 15 to 19 that interact with the music engine SKME of layer III. Information flow between the software and hardware components of layers I to IV is represented in Figure

1 by arrow-heads on dotted-line interconnections, whereas arrow-heads on solid lines indicate an act of creation; for example, information in the composed-notes buffer 11 is used by the conductor 12 which is created by the soundscape 8. The soundscape is a runtime object, only one of which typically exists within the SKME.

The applications layer I determines the look, feel and physical instantiation of the music engine SKME. Users can interact with the music/audio/compositional engine SKME through web applications 1, or through desktop computer applications 2 such as those marketed by the Applicants under their Registered Trade Mark KOAN as KOAN PRO and KOAN X; the music engine SKME may itself be such as marketed by the Applicants under the Registered Trade Mark KOAN. Interaction with the engine SKME may also be through applications on other diverse platforms 3 such as, for example through mobile telephones or electronic toys. All applications 1 to 3 ultimately communicate with the music engine SKME via the API 6 which protects the internals of the music engine SKME from the outside world and controls the way in which the applications can interact with it. Typically, the instructions sent to the API 6 from the applications 1 to 3 consist of commands that instruct the music engine SKME to carry out certain tasks, for example starting the composition and playback, and changing the settings of certain parameters (which may affect the way in which the music is composed/played). Depending on the needs of the individual applications, communication with the API 6 may be direct or via an intermediate API. The SKME according to the present invention resides in one or more servers, and the music or sound generation is effected at the server, for transmittal to one or more remote clients.

The music engine SKME, which is held in memory within the system, comprises eight main components 7 to 14. Of these, SSFIO (SSEYO file input/output) 7, which as the name suggests is for file input/output, holds a description of the parameters, rules and their settings used by algorithms within the engine, to compose. When the engine SKME is instructed via the API 6 to start composition/playback, a soundscape 8 is created in memory and this is responsible for creating a composer 10, conductor 12 and all the individual compositional objects 9 relating to the description of the piece as recorded in the SSFIO 7. The compositional objects are referred to by the composer 10 (which runs in a background loop) to decide what notes to compose next. The composed notes are stored in a number of buffers 11 along with a time-stamp which specifies when they should be played. The conductor 12 keeps time, by receiving accurate time information from a timer device 19 of level IV. When the current time exceeds the time-stamp of notes in the buffers 11, the relevant notes are removed from the buffers 11 and the information they contain (such as concerning pitch, amplitude, play time, the instrument to be used, etc.) is passed to the appropriate rendering objects 13. The rendering objects 13 determine how to play this information, in particular whether via a MIDI output device 17, or as an audio sample via an audio-out device 18, or via a synthesiser engine 14 which generates complex wave-forms for audio output directly, adding effects as needed.

The hardware devices layer IV includes in addition to the devices 17 to 19, a file system 15 that stores complete descriptions of rules and parameters used for individual compose/playback sessions in the system; each of these descriptions is stored as an `SSfile', and many of these files may be stored by the file system

15. In addition, a MIDI in device 16 is included in layer IV to allow note and other musical-event information triggered by an external hardware object (such as a musical keyboard) to be passed into the music engine SKME and influence the composition in progress.

The system can be described as having essentially two operative states, one, a `dynamic' state, in which it is composing and the other, a `static' state, in which it is not composing. In the static state the system allows modification of the rules that are used by the algorithms to later compose and play music, and keeps a record encapsulated in the SSFIO 7, of various objects that are pertinent to the description of how the system may compose musical works. The system is also operative in the dynamic state to keep records of extra objects which hold information pertinent to the real-time composition and generation of these works. Many of these objects (the compositional objects 9 for example) are actual instantiations in memory of the descriptions contained in the SSFIO 7. Modification of the descriptions in the SSFIO 7 via the API layer II during the dynamic state, results in those modifications being passed down to the compositional objects 9 so that the real-time composition changes accordingly.

Figure 2 shows a breakdown of the SSFIO 7 into its constituent component objects which exist when the system is in its static and dynamic states; the system creates real-time versions of these objects when composing and playing. In this respect, SSfiles 20 stored each provide information as to `SSObject(s)' 21 representing the different types of object that can be present in the description of a work; these objects may, for example, relate to piece, voice, scale rule, harmony rule, rhythm rule. Each of these objects has a list of `SSFparameters' 22 that describe it; for example, they may relate to tempo, instrument and scale root. When an SSfile 20 is loaded into the music engine SKME, actual instances of these objects 21 and their parameters 22 are created giving rise to `SSFObjectInstance' 23 and `SSFParameterInstance' 24 as illustrated in Figure 2.

Referring again to Figure 1, the user interacts with the system through applications 1 to 3 utilising the services of the API 6. The API 6 allows a number of functions to be effected such as `start composing and playing', `change the rules used in the composition', `change the parameters that control how the piece is played' including the configuration of effects etc.

The rules and other parameters affecting composition (for example, tempo) within the music engine SKME are defined in memory, specifically within the SSFIO 7, and its real-time instantiation of the compositional objects 9. Use of rules and parameters within the music engine SKME form part of the continual compositional process for other voice objects within the system. Figure 4 illustrates this more general process based on examples of scale and harmony rules shown at (1) and (2) respectively.

Referring to Figure 3, the scale rule is illustrated at (1) with shaded blocks indicating a non-zero probability of choosing that interval offset from a designated scale root note. The larger the shaded block, the greater the probability of the system choosing that offset. Thus, for this example, the octave Ove, major third M3 and fifth 5 are the most likely choices, followed by M2, 4, M6 and M7; the rest will never be chosen. Sequences that may be generated by the system from this are shown below the blocks, and in this respect the octave has been chosen most often followed by the major third and the fifth. With the scale root set in the system as C, the resulting sequence of notes output from the system in this example are C,E,C,D,G,A,E,D,C,G,E,B,C,F, as illustrated at (1) of Figure 3.

The harmony rule defines how the system may choose the pitches of notes when other notes are playing, that is to say, how those pitches should harmonise together. In the example illustrated at (2) of Figure 3, only the octave and major second are indicated (by shading) to be selected. This means that when the pitch for a voice is chosen, it must be either the same pitch as, or a major second from, all other notes currently being played.

For the purpose of further explanation, consideration will be given to the example represented at (3) of Figure 3 involving three voice objects V1-V3. The rhythm rules applicable to the voice objects V1-V3 in this example, give rise to a generated sequence of notes as follows: voice V1 starts playing a note, then voice V2 starts playing a note, then voice V3 starts playing a note, and then after all notes have ended, voice V2 starts playing another note, followed by voice V1 and then voice V3. With this scenario, the note from voice V2 must harmonise with that of voice V1 and the voice V3 note must harmonise with that of voice V2. If in these circumstances the voice V1 is, as illustrated by bold hatching, chosen with a pitch offset of a fifth from the scale root, the pitch for voice V2 must either be the same as (Ove) or a major second above (M2) the fifth. In the case illustrated, it is chosen to be the same, and so the fifth is chosen too. When voice V3 starts playing it must harmonise with both voices V1 and V2, so the pitch chosen must be the same as, or a major second above that of voices V1 and V2. As illustrated, the system chooses voice V3 to be a major second above, therefore giving pitch offset M6 from the scale root.

After voice V3 all notes end, and the next note begins, as illustrated at (4) of Figure 3 with voice V2. This next note by voice V2 is governed by the next-note rule used by voice V2, and the last note played by voice V2. According to this rule, the system chooses pitch offset M2 for voice V2, and then harmonises voices V3 and V1 with it by choice of a major second for both of them. With the scale root set in the system to C, the entire generated sequence accordingly follows that indicated at (5) of Figure 3, where `S' denotes a note starting and `E' a note ending.

Thus, when sequences are generated in response to an external trigger, the actual pitches and harmonisation of that sequence is determined by the composer 10 using several items of information, namely: (a) the note-control sub-pattern operational at that moment; (b) the scale, rhythm, harmony and next-note rules depending upon the type of the note-control subsequence; and (c) any piece-level rules which take into account the behaviour of other voices within the piece.

When the music engine SKME is in dynamic (i.e. composing and playing) mode, it typically contains a number of voice compositional objects 9. The composer 12 composes a sequence of notes for each of these and makes sure they obey the various rules. The process involved is illustrated in the flow diagram of Figure 4.

Referring to Figure 4, the music engine SKME responds to an external trigger applied at step 51, and the API 6 through step 52 instructs a voice 1 in step 53 to register that it must start a sequence. Voice 1 and the voices 2 to N in step

54, have their own rules, and the composer 10 ensures that the relevant rules are obeyed when utilising any of the voices 1 to N. More particularly, the composer 10 responds in step 55 to the instruction of step 53 for voice 1 to start a sequence, by starting the generative pattern sequence sub-system of Figure 3. This sends note-control sub-sequences to the trigger voice (voice 1 in this example), but the composer 10 makes sure the resulting notes harmonise with the other voices in the piece. The outcome via the conductor 12 in step 56 is played in step 57.

The generative pattern sequence triggered will play forever, or until the system is instructed otherwise. If a sequence control sub-pattern is used to define a generative pattern sequence such that the final note control sub-pattern is one which plays silence (rest notes) in an infinite loop, then when this pattern sequence is selected, the voice will become effectively `inactive' until another trigger is detected. Further triggering events for the same generative pattern sequence may sound different as the process is generative, or since the rules in use by the piece or the scale of the trigger voice, its harmony or next note rules may have changed (either via interaction through the API 6 or via internal music engine SKME changes).

The sounds used to `render' each note, whether from triggered sequences or generative voices may be played either through the MIDI sounds or the samples of the rendering objects 13, or via software of the synthesiser engine 14 which may add digital signal processing effects such as, for example, filter sweeps, reverberation and chorus. The entire process can be used to generate musical event information that is then fed into, and may thus control, other processing units within the system such as synthesiser related units allowing the triggering of generative sound effects. Voices can also be added which make use of the software synthesiser engine 14 to generate non note-based effects such as sound washes and ambient environmental sounds, such as chimes, wind and other organic sounds.

The system operates according to a program which allows the user to modify parameters or create structures representing elements of the composition or sound definitions ('user control information') which can either: (1) affect the current group or solo audio session in progress as soon as the changes can be accommodated; or (2) affect a session exclusive to the user or exclusive to a number of users where the user's input is used to create audio material that can be used independently of the session or added to the session at a later time..

Depending on the implementation, the logic of this application might reside either predominantly on the server, or predominantly on the client. This program could be written in a variety of computer languages (for example, client-side Java, Java Servlets, C++ etc.), which can communicate with the server via for example pipes, streams, text blocks, simple messages, MIDI etc. User control information can be either uploaded dynamically to the sever to affect the session on a ongoing basis or instead as a batch-process. In either case, the general principle is that one or more remote clients are controlling or influencing (either in real time or not) sounds or music generated by a server. The client control could be direct – e.g. by explicitly instructing the server to generate music in the key of C major, having a particular rhythm – or it could alternatively be implicit – for example where the client is controlling something else on the server such as an image, and the features of the image then control or influence the sound. The control information could include the selection of pre-defined templates on the server which may control or constrain the sound generation process in a multiplicity of predefined ways. The client may also send information which instructs the server to set up, modify or delete such templates. Once a user has gone to the trouble of defining a template, it may then simply be referred to in the future by a short command which instructs the server to select it.

The audio session might be rendered immediately via an audio stream (subject to a small latency period), or it might be created in a deferred manner for later delivery (saved to an audio recording within the server file system).

Where the application allows many users to interact together to influence the audio output, the users can influence the shared audio experience through a number of mechanisms, including (but not limited to):

(a) voting;

(b) specific ownership of specific musical objects (or characteristics of objects) — this could allow the application to act as a trading system for buying packs of sound/musical components and then trading them with other users (comparable with POKEMON™) — alternatively, it could allow the application to act as a bulletin board for combining components with those of other users, to form bands;

(c) time-slice based ownership;

(d) payment for the over-riding `privilege' to change a given element; and

(e) consolidation of a number of user's inputs for a specific parameter into a single value.

Some example applications are:

1. WAP phone or PDA used to create personal `musicians' (actually SKME voices all stored on the server along with all other data), sound effects, tunes or tracks. Menus or some other data entry mechanism may be used to design these voices, sound effects, tunes or tracks etc.

2. Recordings (audio MIDI, parametric data, control data for example) of band sessions created on the server. Users in the band are notified when session recordings are available for download. WAP phone connects to server to mail links to desktop computer (so that recordings can be heard), or to play recordings directly on the phone where that phone supports appropriate playback, e.g. of embedded MP3 content.

Additionally, users can also interact with a composition in progress or design SKME components via a standard touch-tone telephone system. When they dial up the telephone server, the service translates touch-tone sounds into data (via the telephone server) and sends this data to the SKME server which interprets this as data to modify compositional parameters. By this method a user can, with a touch-tone telephone, make simple modifications to the SKME compositional processes. The user could also hear the composition in progress on the telephone line. This would happen by the server's streamed or stored audio files (as described below) simply being played down the telephone line to the user. In this way the user does not need a special client-device with the ability to receive and play streams.

Rendering may be server-side and audio streamed to the client. In this regard, one or more servers are used to create one or more generative audio streams, and/or one or more recordings of the audio stream are stored. The audio to be created is dictated by the application logic, which is in turn influenced by the users involved in controlling any one particular server-side output.

Figure 5 illustrates audio generation with rendering at the server and audio streaming to the client. The client 61 interacts with an application 62 and 63 to create music structural components (such as voice rules mentioned above, or more sophisticated musical structures). Communication between the client and server-side in this application takes place via a gateway 64 (such as a WAP server for example). There may be many server-side applications in a one to one correspondence with the number of clients connected to it. The created music structural components are then integrated into a database 65 to be queried by the SKME Server Process 66 to be used as the basis for a time-sequenced collection of multiple users' control information and audio samples from 67 to 68 waiting to be composed and rendered. This rendered audio is streamed to the client starting at the server's current playback point. The client then receives and hears the generated streamed audio from the server as represented at 69.

Instead of the process rendering the audio to an audio file to be streamed, the audio may be sent as a steam of data (or control information) in any other music/audio representation format (such as MIDI data). It would be up to the client-side of the application to convert that data-stream into actual sounds (for example, standard MIDI output device or software synthesiser).

As an alternative to the above, audio generation may be with rendering at the server and deferred file delivery to the client. In this respect, user operations are used to influence the creation of audio by batch processes on the server. The server composition process is deferred, such that as and when suitable server-side CPU power is available, the compositions are rendered and the recorded audio sessions are made available for consumption by the user and/or interested users. The presence of these files is then communicated to the user and/or interested users, via e-mail or some other messaging system. The audio recordings may then be listened to at the convenience of each user. This is illustrated by Figure 6 where the client 71 interacts with an application 72 and 73 to create music structural components (such as SKME voice rules mentioned above, or more sophisticated musical structures). Communication between the client and server sides of this application takes place via a gateway 74 (such as a WAP server for example). There may be many server-side applications in a one to one correspondence with the number of clients connected to it. The created music structural components are then integrated into a database 75 of musical compositions to be generated. These are farmed off to numerous `SKME server processes' 76 to generate each client's audio output. When these are completed they are stored as audio files (for example, in MP3 or WAV formats) on a file store 77.

Again, instead of the SKME Server Process rendering the audio to an audio file, the audio may be stored as a sequence of data (or control information) in any other music/audio representation format (such as MIDI data), ready for transmittal to or downloading by the client.

Furthermore, audio generation may be with rendering at the server and audio played to the user via a telephone.

The user could also hear the composition in progress on the phone line. This would happen by the server's streamed or stored audio files simply being played down the telephone line to the user. In this way the user does not need a special client-device with the ability to receive and play streams.

In a further variation, the server may supply the client with a "rough rendering" of the particular sound or music that has been generated. This can be done relatively rapidly at the server, and, because of the small file size, need not take too much transmission time either. If the user is happy with the sound, he instructs the client to send a message back to the server asking for the sound to be rendered at high resolution. The server complies – either immediately or when it has time – and sends a message to the client to indicate when the high-resolution file is ready for downloading.

An alternative site variant of that approach would be for the server to send control instructions to the client enabling the client itself to do the initial "rough rendering". If the result is acceptable to the user, a message is sent back to the server requesting either control instructions enabling the final high-quality sound to be rendered at the client or, alternatively, a request for the high-quality rendering to be carried out at the server.

A user may be allocated an amount of storage space or a number of entries in the database (either for free or for a fee) and further space or entry slots could then be purchased as required. Mail 78 is then sent to the client to inform him/her that the audio generation is completed and ready to be downloaded.

The file store can also be accessed by the web-server 79 which the user then logs into to download his completed audio file. The user can log into this from any web terminal 80 where he can then hear the audio. At this time the user could, for example, mail the audio file to a friend, or save it to a CD.

If the file was stored in a representational format such as MIDI (or other music/audio representation or control format) It would be up to the client-side of the application to convert that data-file into actual sounds (for example, by playing the file through standard MIDI output device or software synthesiser).

Where users want to interact with the music engine on an individual basis (that is to say, they do not want to create a joint composition with other users), then potentially a large number of high-capacity servers are needed to accommodate the correspondingly large number of composed audio-streams. A single server option might be appropriate when there is only a small number of compositions being generated at any one time, for example in a system where many users are sharing the same audio experience; a group or community session. A user may choose to purchase copies of segments of the recordings of the session, perhaps as a memento. These copies would be owned by the user and could be distributed to the user via CD-ROM or e-mail, for example.

Each generated audio output would be either streamed back to one or more clients, or recorded to backing storage for later consumption by the user and/or interested users where such clients support playback of the output. Output could for example be streamed to multiple clients in the case of the user's component being incorporated in a community composition, or the user could for example, at a time after the interactive session, download the recorded output of his composed component. Alternatively the user could hear (where it is in audio form) his composed component or the community composition over a telephone.

The invention has application for group composition sessions where a style is allowed to develop. This may come about through positive feedback when a majority of users note a particular `style' emerging from the work and try to reinforce it. Thus, each server/server-session may embody its own style. Server-sessions may be allowed to interact by passing rule-sets between them. This can act as a primitive form of communication system. A method of assigning fitness to rule-sets could allow spread of rule-sets (which can be seen as style components) from server session to server session. These rule-sets may well operate at different temporal scales, allowing musical components to travel through compositions, much as viruses spread through populations of hosts.

The invention may be applied to sever-side sound design, that is to say to provision of a service in which a user defines the elements of a composition, tune, mix or sound, through a WAP interface. When the user has finished editing the settings, they submit their editing for batch processing/composition on the server (through a server-side engine). They get SMS mailed when their piece/sound/track/mix is ready and can then dial up a number and listen to it (or download it from a PC etc). This could either have a gaming or creativity slant, and could also be submitted to a server-side real-time session.

As suggested above, provision of a server-side real-time session may be through a service in which the customer dials a number and can listen to a composition being created or changed in real time by the listeners themselves. Each key-press can control certain aspects of the group piece, such as, for example, consolidated general voting when the listener likes what is happening, or perhaps even some direct user control over elements they have submitted during server-side sound design as described in the preceding paragraph. But in general such control would be at a lower level than in server-side sound design, since those creating the elements in server-side sound design are effectively controlling the "space" — so there is a distinct advantage in having a WAP phone and becoming expert at designing the sounds/riffs/tunes etc. When GPRS becomes available, users will be able to engage in server-side design and server-side real-time sessions concurrently.

Server-side sound design has many applications, for example:

(a) WAP musician trading/role-playing game - A WAP or other mobile phone is used to create personal `musicians' (actually SKME voices, all stored on the server along with all other data) — menus are used to design these voices, to define sounds to use etc;

(b) trading system for buying packs of sounds and trading them with other users (comparable with POKEMON™);

(c) bulletin-board system for combining these musicians with those of other users, to form bands;

(d) MP3 or other `sessions' of band recordings created on the server — the users in the band would be mailed when sessions are available for download, and the WAP or other mobile phone connects to server to mail links to desktop computer, so that files can be heard, or to play files directly on the phone where that phone supports playback of e.g. embedded MP3 content;

(e) users can pay to have sessions cut to CD;

(f) whenever a desktop box is used to access a page which contains the file links, a KOAN Plugin is forced to be downloaded;

(g) enhanced on-line instruments for subsequent on-line mixing of sessions or pieces - these would be available only to users who paid for the phone service; users may also pay for increased server priority, longer recordings etc;

(h) the generation of ring tones for mobile phones. As described above, the ring tone could either be sent to the phone as fully-rendered audio, or alternatively by means of a series of control instructions (e.g. a midi-file or in OTA format) allowing the phone to render its own tone. The tone could be either polyphonic or monophonic;

(i) sound-generation at the server being linked to visual displays on the client side. For example, where a number of users are engaged in a multi-user game, within a virtual 3-D world, for example via the Internet, the client instructions to the server may affect what happens visually within the virtual world as well as what happens aurally. The aural and visual aspects may be linked, for example to increase the realism of the virtual environment. For example, each player in a computer game may be represented by a corresponding computer-generated player who is able to move around within the virtual world. The sounds "heard" by that individual, produced by the server and transmitted to the corresponding client, may depend upon the position of the individual within the virtual world. The sound of an explosion could for example vary according to where the individual is located, and in particular how far away within the virtual world the explosion is. Likewise, the user may be supplied with differing ambient or background sounds depending upon the location within the world of the corresponding computer-generated individual. More generally, sounds may be generated by the server and transmitted to one or more clients which depend upon the relative position of objects (not necessarily individuals) within the virtual world. In order to generate or influence the sounds, a user simply uses normal game controls to move objects or individuals around within the virtual world; the server feeds information about the relative virtual positions of the objects into the SKME, and the resultant sounds are generated and/or influenced accordingly without the user needing to control the sounds directly. Generally, then, the audio engine on the server is configured to provide a stream of audio (music and other audio events) to the user that reflects the state of the ongoing user experience within the virtual environment. The audio the user will experience is weighted to reflect the greater significance of local events.

CLAIMS:

1. A system for the composition of audio sequences comprising a server and a generative audio engine on the server controlled by a plurality of remote clients in communication with the server, the server being arranged to transmit a completed composed audio sequence to one or more of the clients.

2. A system as claimed in claim 1 in which the audio sequence is rendered on the server and transmitted in rendered form to the client.

3. A system as claimed in claim 1 in which the audio sequence is transmitted in a parametric or encoded format, for example in MIDI format, and is rendered on receipt by the client.

4. A system as claimed in any one of the preceding claims in which the clients are mobile phones.

5. A system as claimed in claim 4 in which the audio sequence includes a ring tone.

6. A system as claimed in any one of the preceding claims in which the audio sequence includes a musical composition.

7. A system as claimed in any one of the preceding claims in which the audio sequence includes generated non-musical sounds.

8. A system as claimed in any one of the preceding claims in which the audio sequence is streamed in real time to one or more of the clients.

9. A system as claimed in any one of claims 1 to 7 in which the audio sequence is, once generated, held on the server until a client request is received for it to be sent.

10. A system as claimed in claim 7 in which the server informs one or more clients when the completed audio sequence is ready to be sent.

11. A system as claimed in any one of the preceding claims in which the audio engine generates the audio sequence in dependence upon instructions received from a plurality of clients.

12. A system as claimed in any one of the preceding claims in which the audio sequence is generated by the audio engine in real time, with the generation being controlled or influenced by instructions received during generation.

13. A system as claimed in any one of the preceding claims in which the audio engine is controlled by control templates stored at the server, the control templates being changeable by the clients.

14. A system as claimed in any one of claims 1 to 12 in which the audio engine is controlled by control templates stored at the server, the control templates being selectable by the clients.

15. A system as claimed in any one of the preceding claims in which the audio engine is associated at the server with an image, the engine being controlled in accordance with changes in the image effected via instructions received from the clients.

16. A system as claimed in any one of claims 1 to 14 in which the audio engine is associated at the server with a virtual 3-dimensional world, the engine being controlled in accordance with changes in the virtual world effected via instructions received from the clients.

17. A system as claimed in claim 16 in which the said changes are changes in the relative or absolute positions of objects or individuals within the virtual world.

18. A system as claimed in claim 16 or claim 17 in which the audio sequence is representative of the sounds heard by a virtual individual or object within the virtual world.

19. A system as claimed in any one of the preceding claims when dependent upon claim 11 in which each client's instructions control a specific musical object or characteristic of such an object within the audio engine, for example key, rhythm, voice, reverberation, and so on.

20. A system as claimed in any one of the preceding claims when dependent upon claim 11 in which the audio engine creates the audio sequence in dependence upon votes received from the clients.

21. A system as claimed in any one of the preceding claims when dependent upon claim 11 in which instructions from more than one client are combined into one or more parameter values, the parameter value or values controlling the audio engine.

22. A system as claimed in any one of the preceding claims including a plurality of linked servers, each having a generative music engine.

23. A computer game using a system as claimed in any one of the preceding claims.

24. A method for the composition of audio sequences, comprising composing an audio sequence at a central location, under control of one or more clients at remote locations, and transmitting the completed audio sequence to one or more of the clients.

25. A generative audio engine arranged to compose audio sequences under the control of instructions received from remote clients, the engine including means for transmitting a completed audio sequence to one or more of the clients.

26. A computer game using a method as claimed in claim 24.

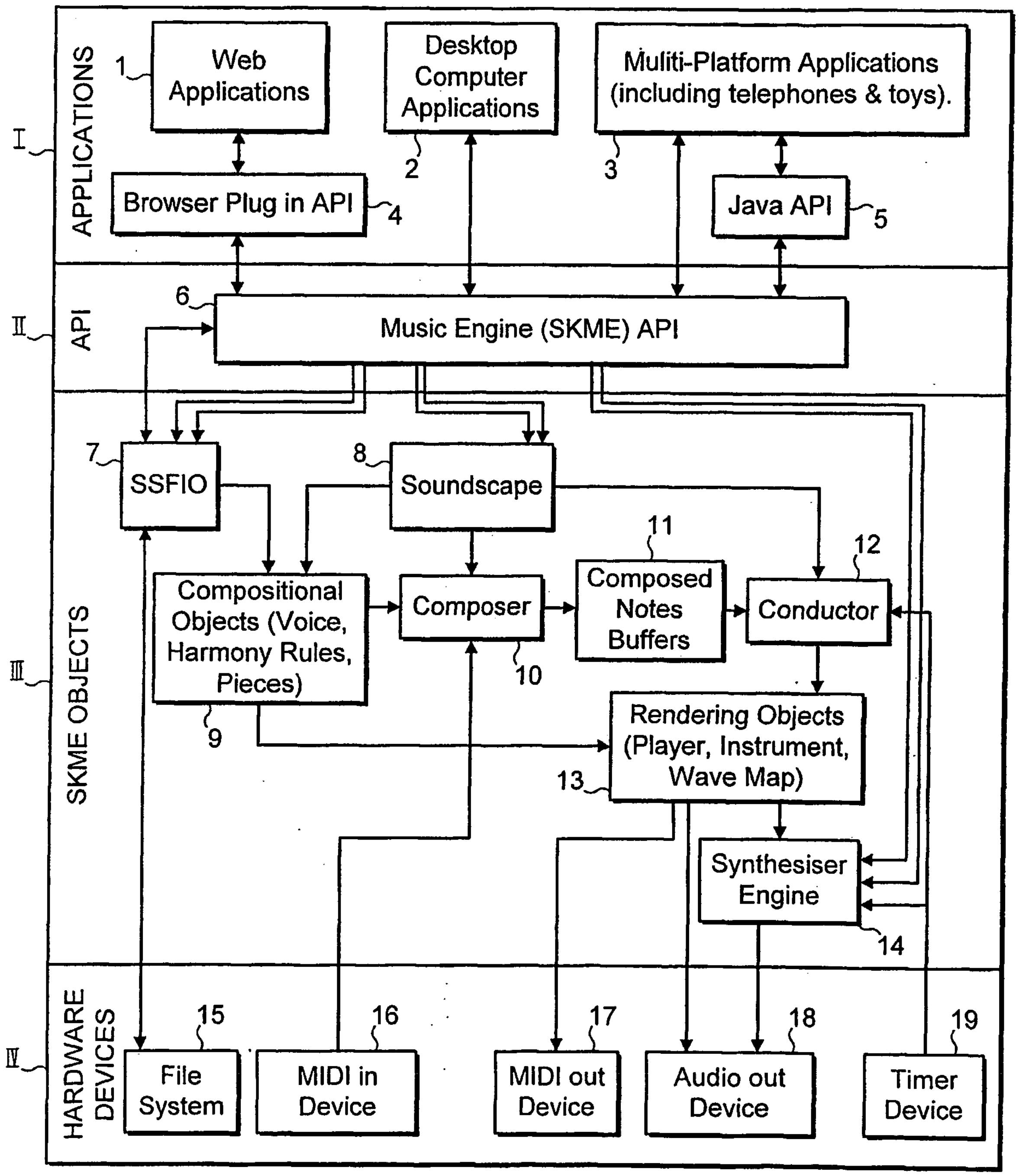

APPLICATIONS
I
1
Web Applications
Desktop Computer Applications
2
Muliti-Platform Applications (including telephones & toys).
3
Browser Plug in API
4
Java API
5
API
II
6
Music Engine (SKME) API
SKME OBJECTS
III
7
SSFIO
8
Soundscape
Compositional Objects (Voice, Harmony Rules, Pieces)
9
Composer
10
11
Composed Notes Buffers
12
Conductor
Rendering Objects (Player, Instrument, Wave Map)
13
Synthesiser Engine
14
HARDWARE DEVICES
IV
15
File System
16
MIDI in Device
17
MIDI out Device
18
Audio out Device
19
Timer Device

FIG. 1

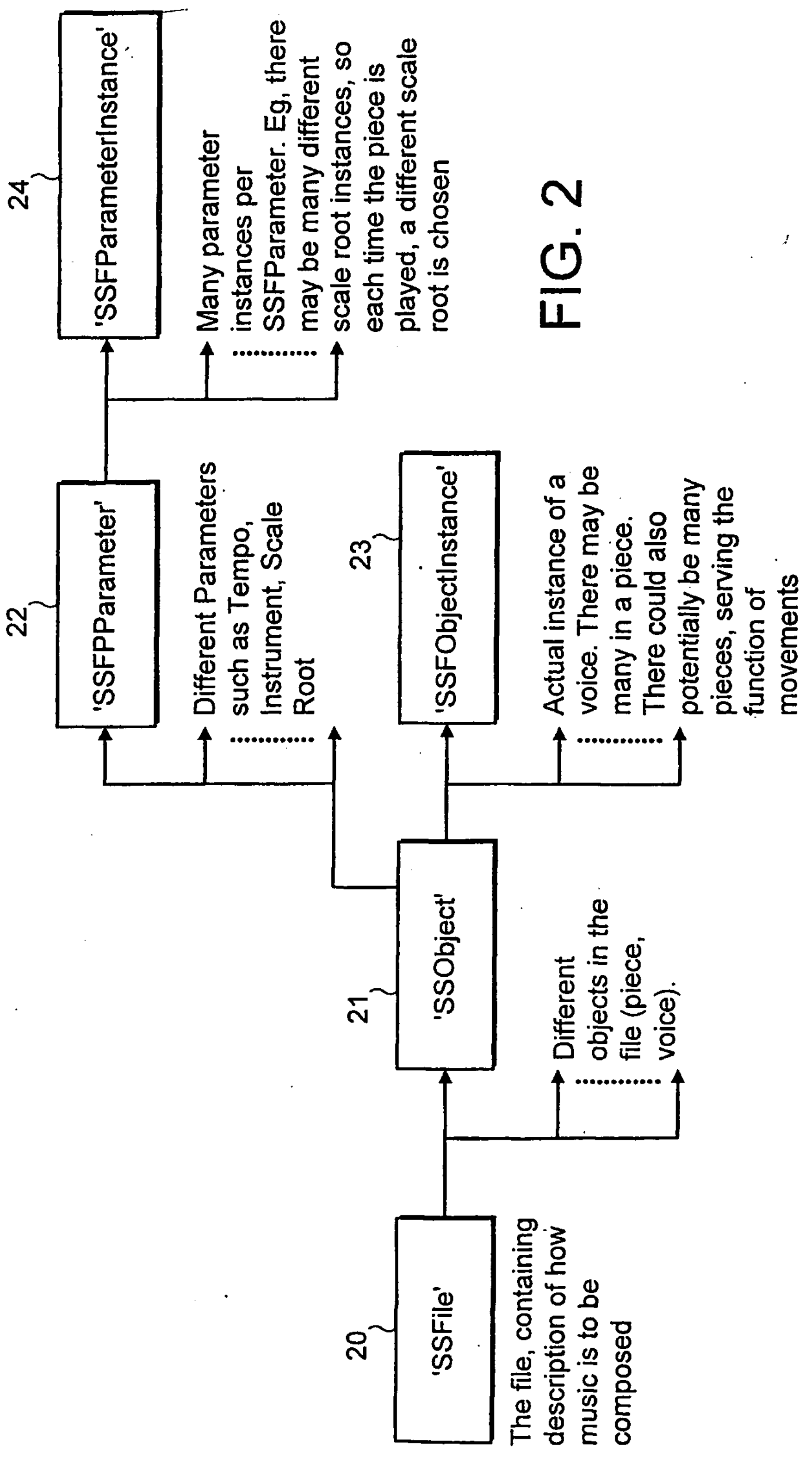
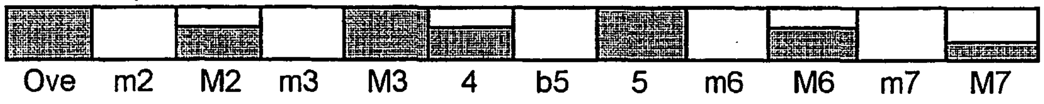

20
'SSFile'
The file, containing description of how music is to be composed
21
'SSObject'
Different objects in the file (piece, voice).
22
'SSFPParameter'
Different Parameters such as Tempo, Instrument, Scale Root
23
'SSFObjectInstance'
Actual instance of a voice. There may be many in a piece. There could also potentially be many pieces, serving the function of movements
24
'SSFParameterInstance'
Many parameter instances per SSFParameter. Eg, there may be many different scale root instances, so each time the piece is played, a different scale root is chosen

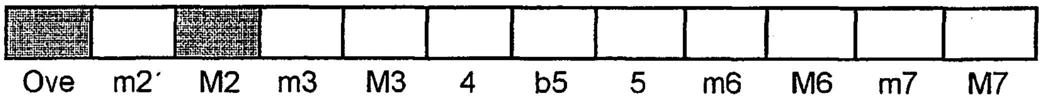
FIG. 2

1 An example Scale Rule:

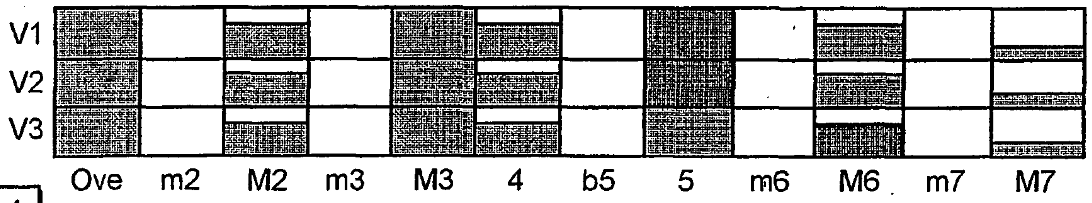


Relative to a scale root, a sequence of notes chosen from this scale rule might be:
Ove, M3, Ove, M2, 5, M6, M3, M2, Ove, 5, M3, M7, Ove, 4
With the scale root set at C, this sequence becomes:
C, E, C, D, G, A, E, D, C, G, E, B, C, F

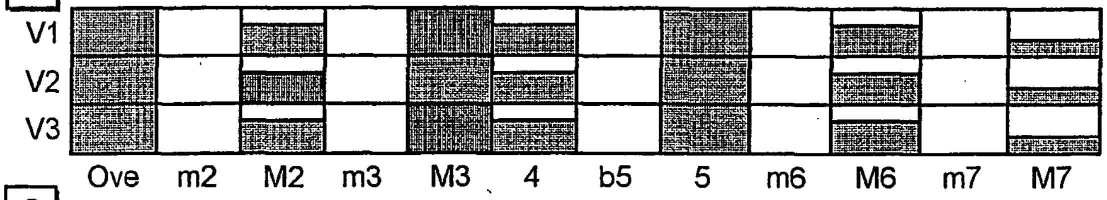
2 An example Harmony Rule:



3



4



5

Time Sequence:
S:V1:G, S:V2:G, S:V3:A,
E:V1. E:V2, E:V3
S:V2:D, S:V3:E, S:V1:E
E:V2, E:V3, E:V1

FIG. 3

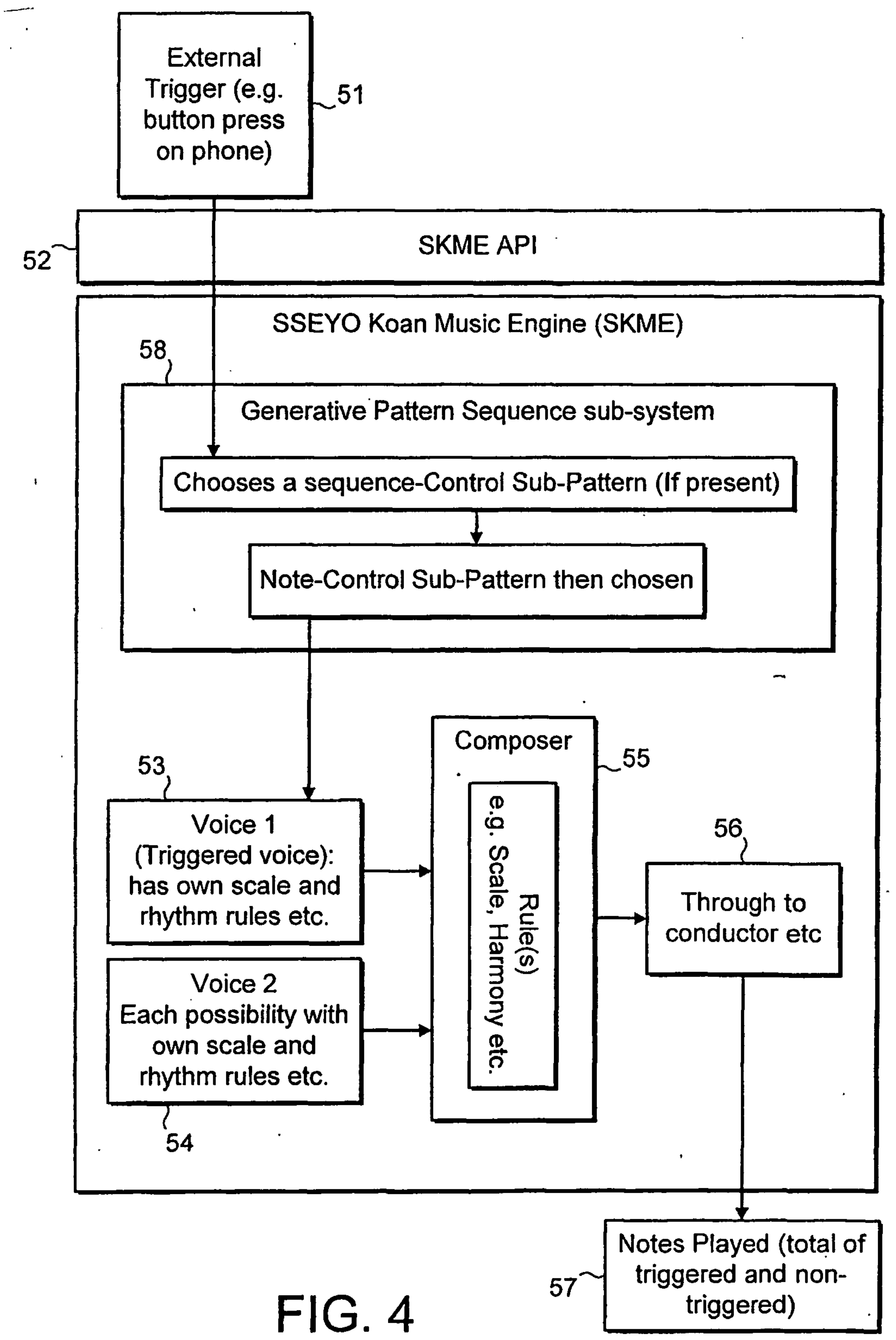
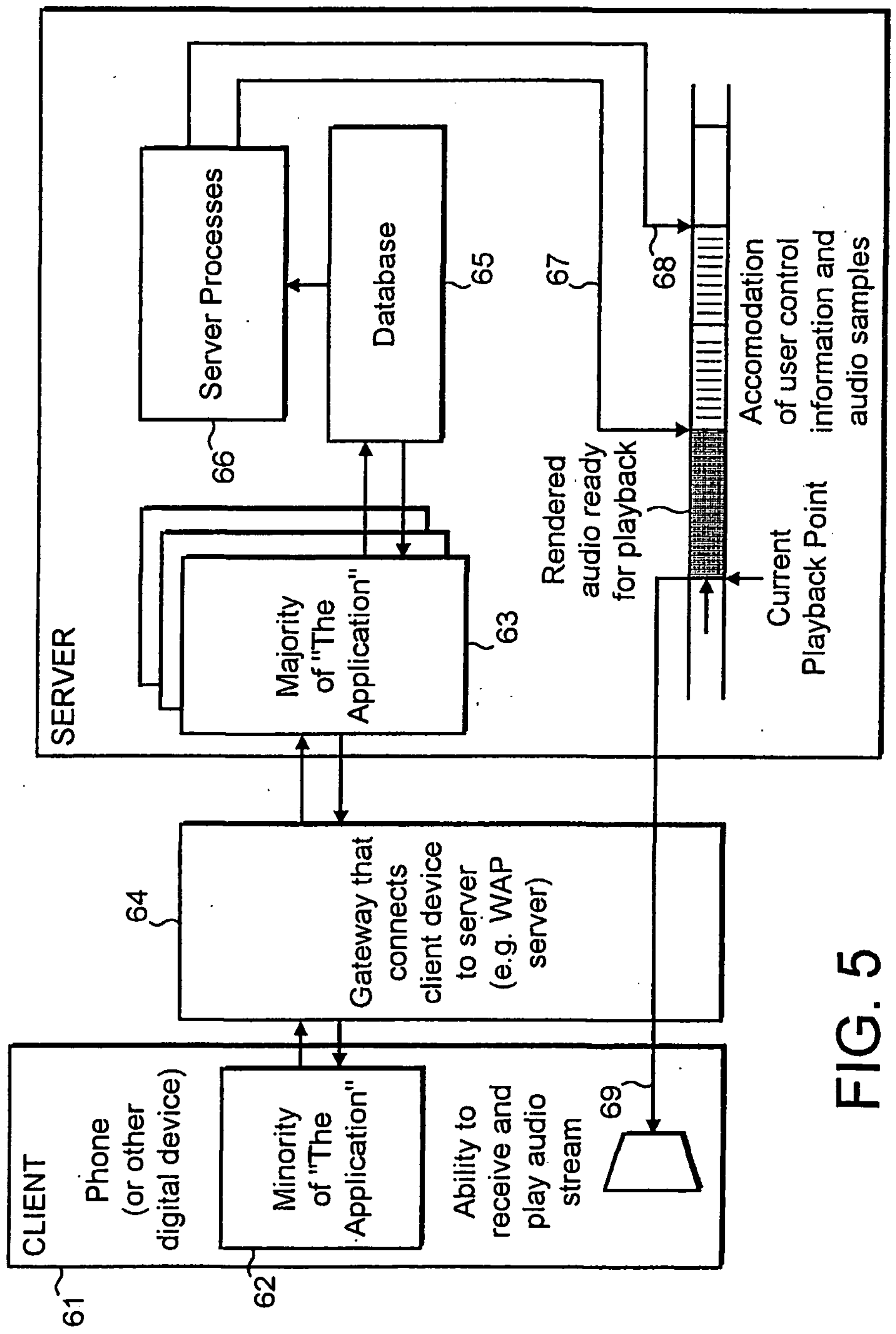

External Trigger (e.g. button press on phone)
51
52
SKME API
SSEYO Koan Music Engine (SKME)
58
Generative Pattern Sequence sub-system
Chooses a sequence-Control Sub-Pattern (If present)
Note-Control Sub-Pattern then chosen
53
Voice 1 (Triggered voice): has own scale and rhythm rules etc.
Voice 2 Each possibility with own scale and rhythm rules etc.
54
Composer
55
Rule(s) e.g. Scale, Harmony etc.
56
Through to conductor etc
57
Notes Played (total of triggered and non-triggered)

FIG. 4



CLIENT
Phone (or other digital device)
Minority of "The Application"
Ability to receive and play audio stream
61
62
69
Gateway that connects client device to server (e.g. WAP server)
64
SERVER
Majority of "The Application"
63
Server Processes
66
Database
65
67
68
Rendered audio ready for playback
Current Playback Point
Accomodation of user control information and audio samples

FIG. 5

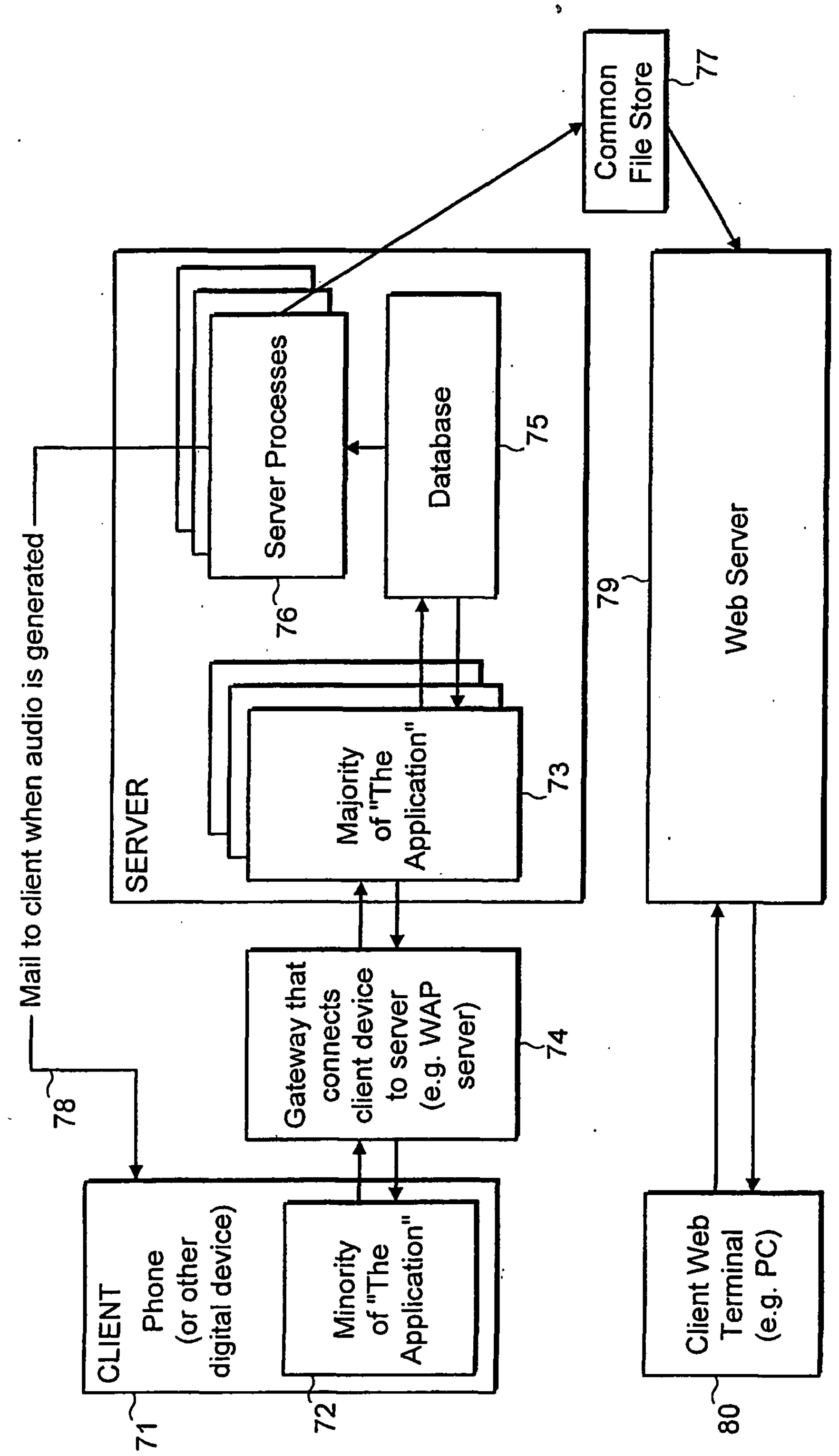

CLIENT
Phone (or other digital device)
Minority of "The Application"
71
72
Gateway that connects client device to server (e.g. WAP server)
74
SERVER
Majority of "The Application"
73
Database
75
Server Processes
76
Mail to client when audio is generated
78
Common File Store
77
Web Server
79
Client Web Terminal (e.g. PC)
80

FIG. 6